Chapter 18 Macro User's Manual

This document is the user's manual of macro module, which describes syntax, usage, and programming methods of macro commands.

The document includes the following chapters:

Macro Description
Macro Usage Description
Macro Commands and PLC Communication (Local Bit, Local Word)
Macro Operation Instruction
Notes about Using Macro
Compiling Error Message
Source Code Examples

- Macro Description

1. Constants and Variables

a. Constants

(1) Decimal constant

(2) Hexadecimal constant

(3) ASCII code (character constant)

(4) Boolean: True (not zero), False (zero)

b. Variables

(1) Naming rules

A variable must start with an alphabet and no longer than 32 characters.

(2) Variable types

char.	Character (8 bit) variable
bool	Boolean(1 bit) variable
short	Short integer(16 bit) variable
int	Double word (32 bit)variable
float	Floating point (32 bit) variable

c. Operator

(1) Assignment operator

Assignment operator：=

(2) Arithmetic operators

Addition ： +

Subtraction ： -

Multiplication ： *

Division ： /

Modulo Division： %

(3) Comparison operators

Less than ： <

Less than or equal ： <=

Greater than ： >

Greater than or equal ： >=

Equal ： ==

Not equal ： <>

(4) Logic operators:

Conditional AND ： and

Conditional OR ： or

Exclusive OR : xor

Boolean NOT : not

(5) Bitwise and shift operators:

(a) Shift operators

Left shift : <<

Right shift : >>

(b) Bitwise operators

Bitwise AND : &

Bitwise OR : |

Bitwise XOR : ^

Bitwise complement : ~

2. Priority of operators

The process order of many operators within an expression is called the priority of operators.

a. Priority of the same kind of operator (From left to right, from up to low)

Arithmetic operator: ^ → (* , /) → (mod) → (+ , -)

Shift operator: From left to right within the expression

Comparison operator: From left to right within the expression

Logic operator: Not → And → Or → Xor,

b. Arithmetic operator is prior to Bitwise operator

Bitwise operator is prior to Comparison operator

Logic operator is prior to Assignment operator

3. Array

Only support fixed length, 1-D array which is:

1-D array: Array_Name [Array_Size]

The array size can be integer which from 0 to 4294967295

Minimum of array index = 0

Maximum of array index = Array size – 1

Example : Array[MAX] MAX = 100

Minimum of array index = 0

Maximum of array index = 99 (100 – 1)

4. Expression

a. Operation object

(1) Constants

(2) Variables

(3) Array

(4) Function

b. Components of expression

An expression is combined operation objects with operators by following specific rules.

5. Statement

a. Definition statement

(1) type name: Define the type of name

Example: int a, Define variable a as an integer

(2) type name[constant]: Define the type of array name

Example: int a[10], Define variable a as a 1-D array of size 10

Assignment statement

The form is ： Variable = Expression

Example: a = 2

b. Logic statement and branches

(1) One-line format

if Condition **then**

[Statements]

end if

Example:

if a == 2 **then**

b = 1

else

b = 2

end if

(2) Block format

if Condition **then**

[Statements]

[**else** [**if** Condition – n **then**

[Else_If_Statements] ….

[**else**

[Else_Statements]]

]]

end if

Example:

if a == 2 **then**

b = 1

else if a == 3

b = 2

else

b = 3

end if

Syntax description

Condition	Necessary. This is a control statement. It will be FALSE when the value of condition is 0; and will be TRUE when the value of condition is 1.
Statements	It is optional in block format statement but necessary in one-line format without ELSE. The statement will be executed when the condition is TRUE.
Condition – n	Optional. See Condition.
Else_If_Statements	Optional in one-line or multi-line format statement. The **else if** statement will be executed when the relative Condition – n is TRUE.
Else_Statements	Optional. The **else** statement will be executed when Condition and Condition—n are both FALSE.

c. Looping control

(1) for–next Statement

Use this for fixed execution counts. To means increase by step while down means decrease by step.

for Counter = Start to end [step Step]
[Statements]
next [Counter]

for Counter = Start down end [step Step]
[Statements]
next [Counter]

Example:

```
for a = 0 to 10 step 2
    b = a
next a
```

Syntax description

Counter	Necessary. The counter of looping control. It can be integer or character.
Start	Necessary. The initial value of Counter.
End	Necessary. The end value of Counter.
Step	Optional. The increment/decrement step of Counter. It can be integer and can be omitted when value is 1.
Statements	Optional. Statement block between For and Next which will be executed fixed counts.

(2) while – wend statement

Loop controlled by Condition. When Condition is TRUE, the statements will be executed repetitively until the condition turns to FALSE.

```
while Condition
    [statements]
wend
```

Example:

```
while a == 2
    b = b + 1
    GetData(a, “Local HMI”, LB, 5, 1)
wend
```

Syntax description

Condition	Necessary. Logic expression which control the execution of statements.
Statements	Optional. Statement block. The statement will be executed when the condition is TRUE.

(3) break

Used in looping control or select statement. It skips immediately to the end of the statement.

(4) continue

Used in looping control statement. It quits the current iteration of a loop and starts the next one.

(5) return

To stop executing the current method.

Reserved keywords:

The following keywords are reserved for Macro which can not be used in function name, array name, or variable name.

+ , - ,* , / ,^, mod, >= ,>, < ,<=, <> , == , And, Or, Xor, Not, <<, >>, = , & , |, ^,~,If ,Then,Else,EndIf,Select ,Case ,For, To, Down Step, Next, while, wend break, continue, return.

- Macro usage description

1. Local variables and global variables

a. Local variables: Its value remains valid only within a specific statement.

b. Global variables: Its value always remains valid after declaration.

When local variable and global variable have the same declaration name, only the local variable will be valid.

2. Variable and constant initialization

a. Variable initialization

(1) Initialize a value of variable in the declaration statement directly.

Example: int h = 9

(2) Use assignment operator to initialize a value after declaration.

Example: temp = 9

(3) Array initialization

Format: int g[10] = { 1,2,3, , 10 }

The initial values are written within the {} and divided by comma (,). These values are assigned orderly from left to right starting from array index=0.

b. Constans.

Macro supports:

(1) Decimal integer constant

(2) Hexadecimal integer constant: start with 0x

(3) Character constant,

(4) Boolean constant: True / False,

3. Boolean variables and Boolean expressions

a. Boolean variables:

True or False. Not zero value means TRUE while zero value means FALSE.

b. Boolean expressions:

The value of Boolean expression is not zero mean TRUE.

The value of Boolean expression is zero mean FALSE.

4. Declaration statement

a. Declaration outside a function is a global variable declaration.

b. Declaration inside a function is local variable declaration. This declaration must at the very beginning of a statement within a function. Other statements before declaration statements will cause compiler error.

For example：

```
macro_command main( )
char i

i = 9//Assign statement within declaration area causes compiler error
int g[10]

for g[2] = 0 to 2
g[3] = 4
next g[2]
end macro_command
```

5. Function call and passing parameters

a. Function define

The format of function statement is:

sub Type FunName(Type1 var1, Type2 var2, …, TypeN varN)

……..

return ret

end sub

Type is the return datae type, FunName is function name, Type1~TypeN is the parameter variable type, var1~ varN is parameter, ret is the return data

For example：

```
sub int func(int i)
        int h

            h = i + 10
        …..
        return h
```

end sub

b. Function call

A function must be defined before its execution. Otherwise, a compiler error ‘Function not defined’ will occur.

For example：

```
macro_command main()
    int    i
    i = Func(i) // call an undefined function causes compiler error
end macro_command
```

c. Passing parameters

(1) Passing by value through local variable.

(2) Through the same global variables

6. Main Function

Macro must has one and only one main function which is the execution start point of Macro. The format is:

```
macro_command Function_name()



end macro_command
```

- Macro command and PLC communication (LocalBit, LocalWord):

Usage：Communicate with PLC through a function library
In the command program, Macro can communicate with data in the PLC. The function GetData(…) can receive data from the PLC through EasyView. The function SetData(…) can set data to the PLC through EasyView. The Macro command handles the communication details.

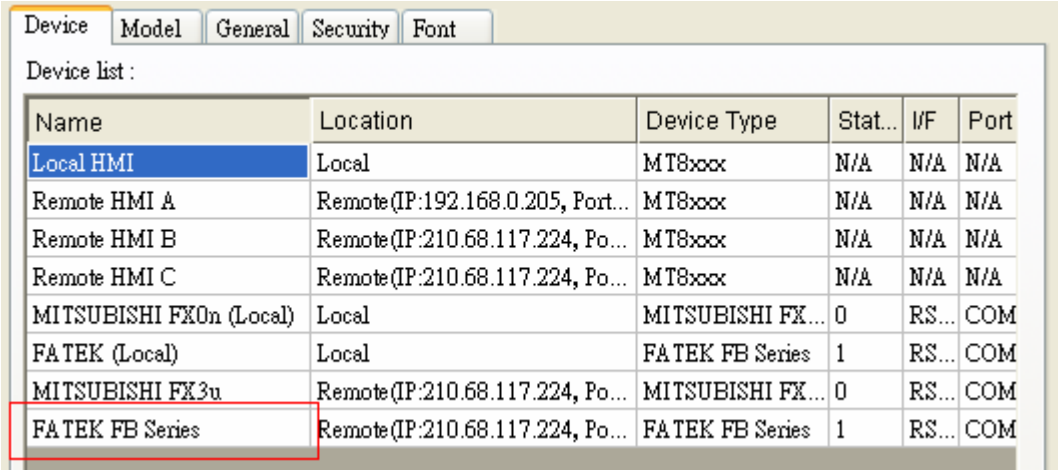
1. GetData(Supported data types: DestData,
   char* szPLCName,
   char* szDeviceType,
   int nAddress,
   int nDataCount)

Description
Get data from PLC
Parameters：

DestData	The address of data to get
szPLCName	PLC name
szDeviceType	PLCtype and encoding method of PLC address
nAddress	The address of PLC
nDataCount	Number of data

Return value:
None

szPLCName
Set the PLC operation object, identify the plc name by use the double quotation marks, those name have been defined in the Device List of System rameter.Example:”FATEK FB Series”, See the following graph, If use the name never been defined in the Device List,will will cause compiler error.
The name of HMI is fixed as “Local HMI”.



strDeviceType Format

AAA_BBB

AAA is the register name in the PLC. Example: LB or LW,BBB means the format data (BINor BCD).

For example: if strDeviceType is LB_BIN, It means the register is LB and the format is BIN.

If use BIN format, ”_BIN” can be ignored, Example: LW_BIN is equal toLW, they both mean the register is LW and the format is BIN.

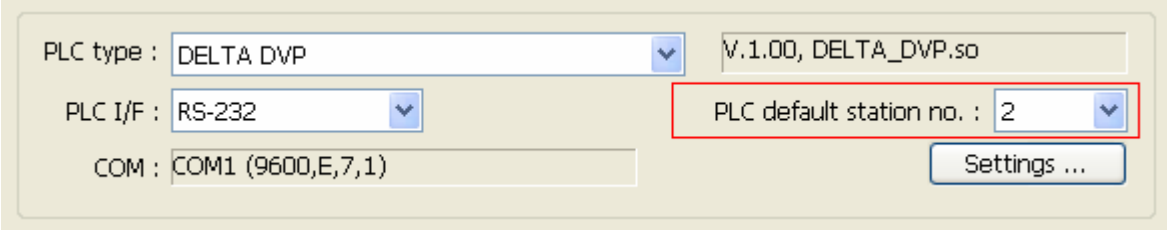
NAddress format

N#AAAAA

N means the station number of PLC, range is from 0 to 255. If use the default stationnumber in system parameter, ’N#’ can be canceled, ；AAAAA is the address ofPLC register.

For example: if strAddress is 2#10, It means the station number of plc is 2,the 表 address of plc register is 10. So the function GetData(a, “DELTA DVP”, M, 2#10, 1) means that read the data in the address M10 of “DELTA DVP” No.2 PLC.

If strAddress is, ’N#’ is canceled, Now It will use the default station number of system parameter.See the following graph, now the default number is 2。



For example :

bool a
bool b[30]
short c
short d[50]
int e
int f[10]
double g[10]

// read the state of LB2, and save in variable a
GetData(a, “Local HMI”, LB, 2, 1)

// Read the total 30 states of LB0~LB29, and save in variables b[0]~b[29]
GetData(b[0], “Local HMI”, LB, 0, 30)

// Read one word data from LW2, and save in variable c
GetData(c, “Local HMI”, LW, 2, 1)

// Read total 50 word datas from LW0~LW49, and save in variables d[0]~d[49].
GetData(d[0], “Local HMI”, LW, 0, 50)

// Read one double word from LW6 ,and save in variable e
// note：the type of e is int
GetData(e, “Local HMI”, LW, 6, 1)

// read total 20 word data, and save in variables f[0]~f[9]
// note：the type of f[10] is int
// f[0] save the data of LW0~LW1, f[1] save the data of LW2~LW3, the rest may be deduced by analogy, GetData (f[0], “Local HMI”, LW, 0, 10)

// read one float data from LW2, size is double word, and save in the variable f

GetData(f, “Local HMI”, LW, 2, 1)

2. SetData (Supported data types: DestData,

char*	szPLCName,
char*	szDeviceType,
int	nAddress,
int	nDataCount)

Description

Send data to PLC,data can be inputted by filling a dialog.

Parameters：

DestData	The address of data to set
szPLCName	PLC Name
szDeviceType	PLCtype and encoding method of PLC address
nAddress	The address of PLC
nDataCount	Number of data

Return value

None

For example :

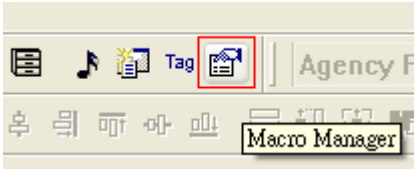
```
int i
bool a = True
bool b[30]
short c = False
short d[50]
int e = 5
int f[10]

for i = 0 to 29
b[i] = true
next i

for i = 0 to 49
d[i] = i * 2
next i
```

```
for i = 0 to 9
f[i] = i * 3
next i

//      set the state of LB2
SetData(a, “Local HMI”, LB, 2, 1)

//    set states of LB0~ LB29
SetData(b[0], “Local HMI”, LB, 0, 30)

//    set the data of LW2
SetData(c, “Local HMI”, LW, 2, 1)

//    set datas of LW0~LW49
SetData(d[0], “Local HMI”, LW, 0, 50)

//    set the data of LW6~LW7
//    note：the type of e is int
SetData(e, “Local HMI”, LW, 6, 1)

//    set datas of LW0~LW19
SetData(f[0], “Local HMI”, LW, 0, 10)
```

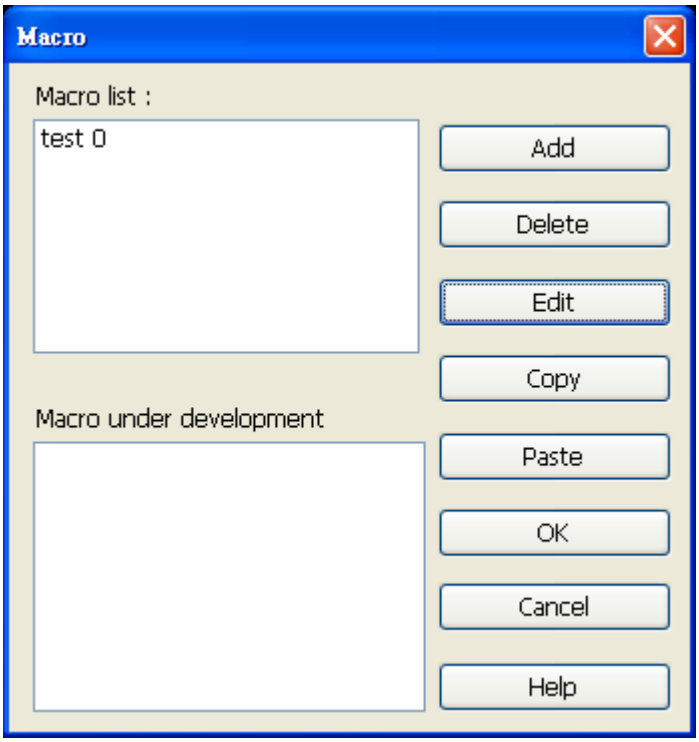
- Macro operation manual

1. Macro programming can be divided into three steps:

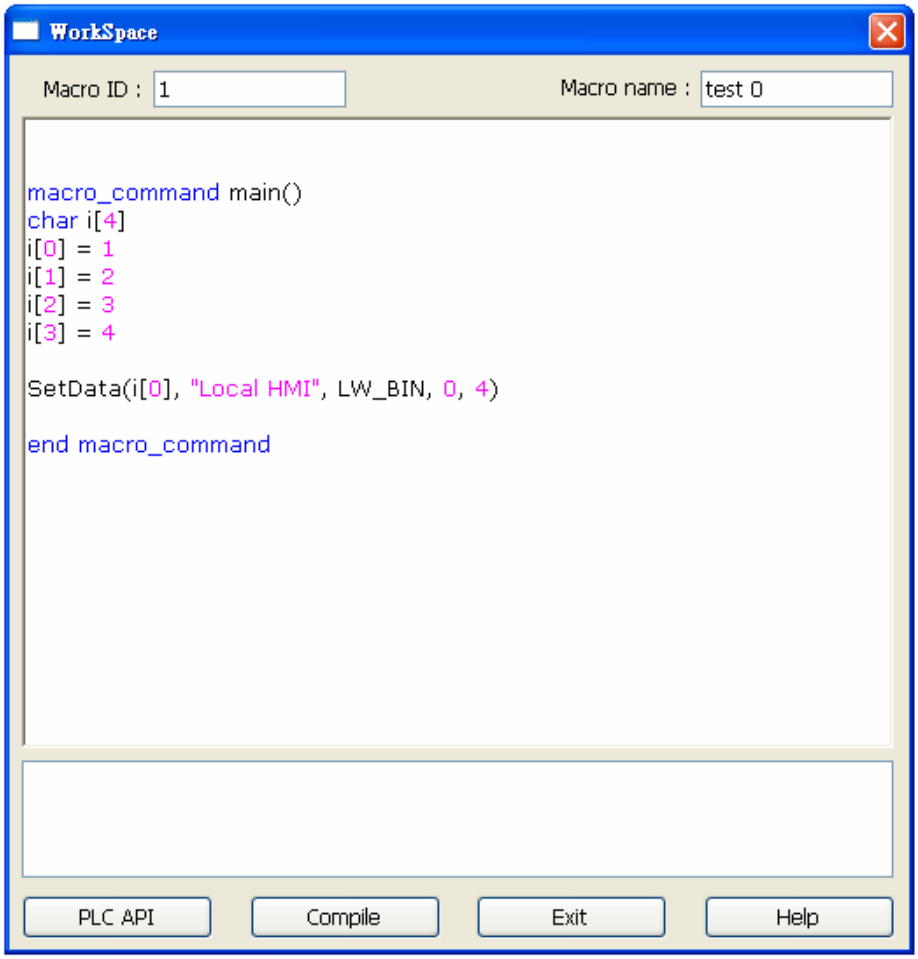
Step 1：click the first icon in the Macro tool box of EasyBuilder 8000



Step 2： Each Macro can be copied, deleted or edited in MacroControlDlg dialog. The source code of Macro can be edited by opening MacroWorkSpaceDlg dialog.



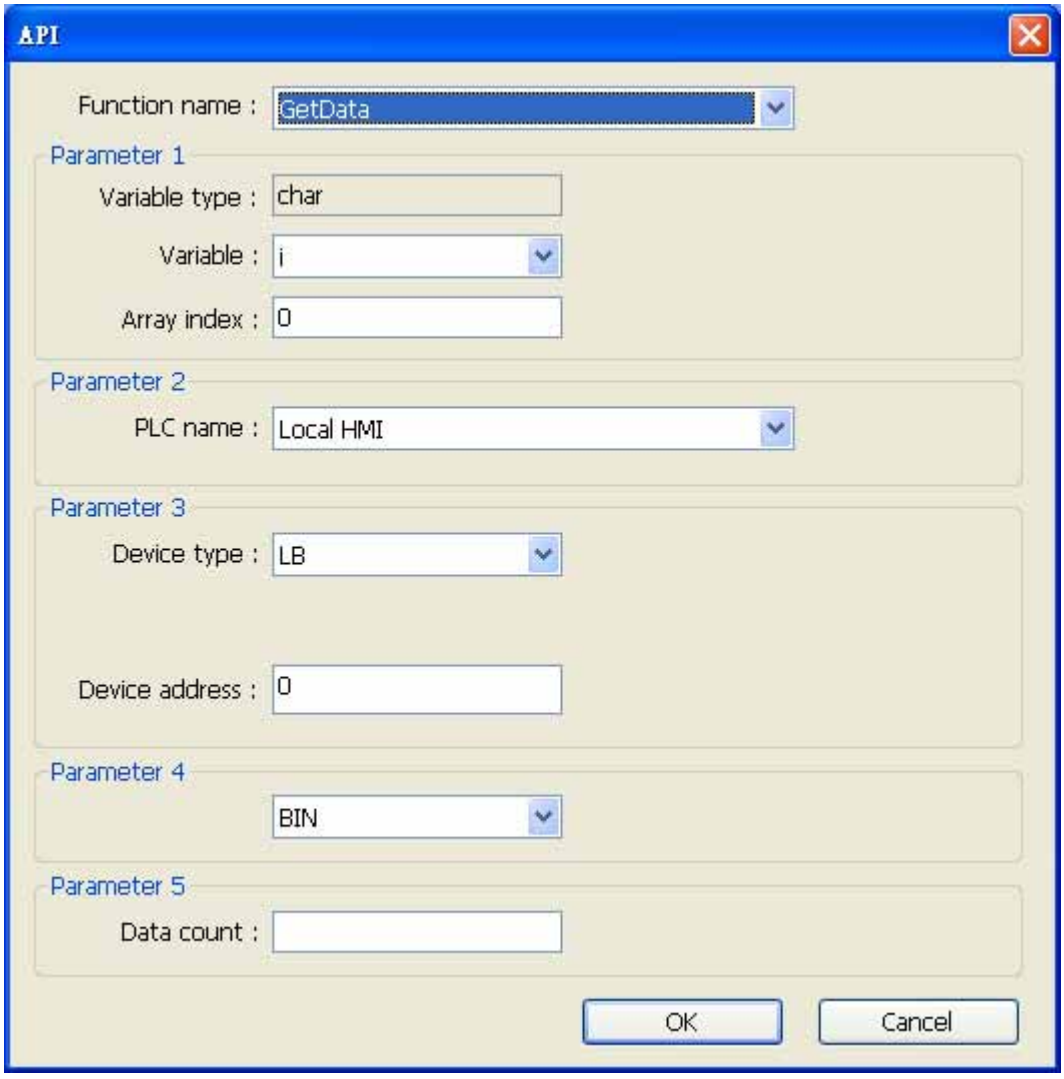
Step 3: Editing the source code of the Macro. Make sure the name and number of the Macro are correct. Compile the Macro and fix the error message.



2. Editing the communication source code of Macro:

a. Input

Step 1： Enter the keyword “Insert” in the proper position. Or by moving the cursor to the proper position and push [PLC API] button. There will appear a dialogue.



Step 2：Select functions and parameters of the library in Library Editing Dialog. Push button “OK” to enter this sub-function; push button “Cancel” to abort this sub-function.

b. Edit

Move the cursor onto the modifying position to modify it. Follow the detail Step2

of Input.

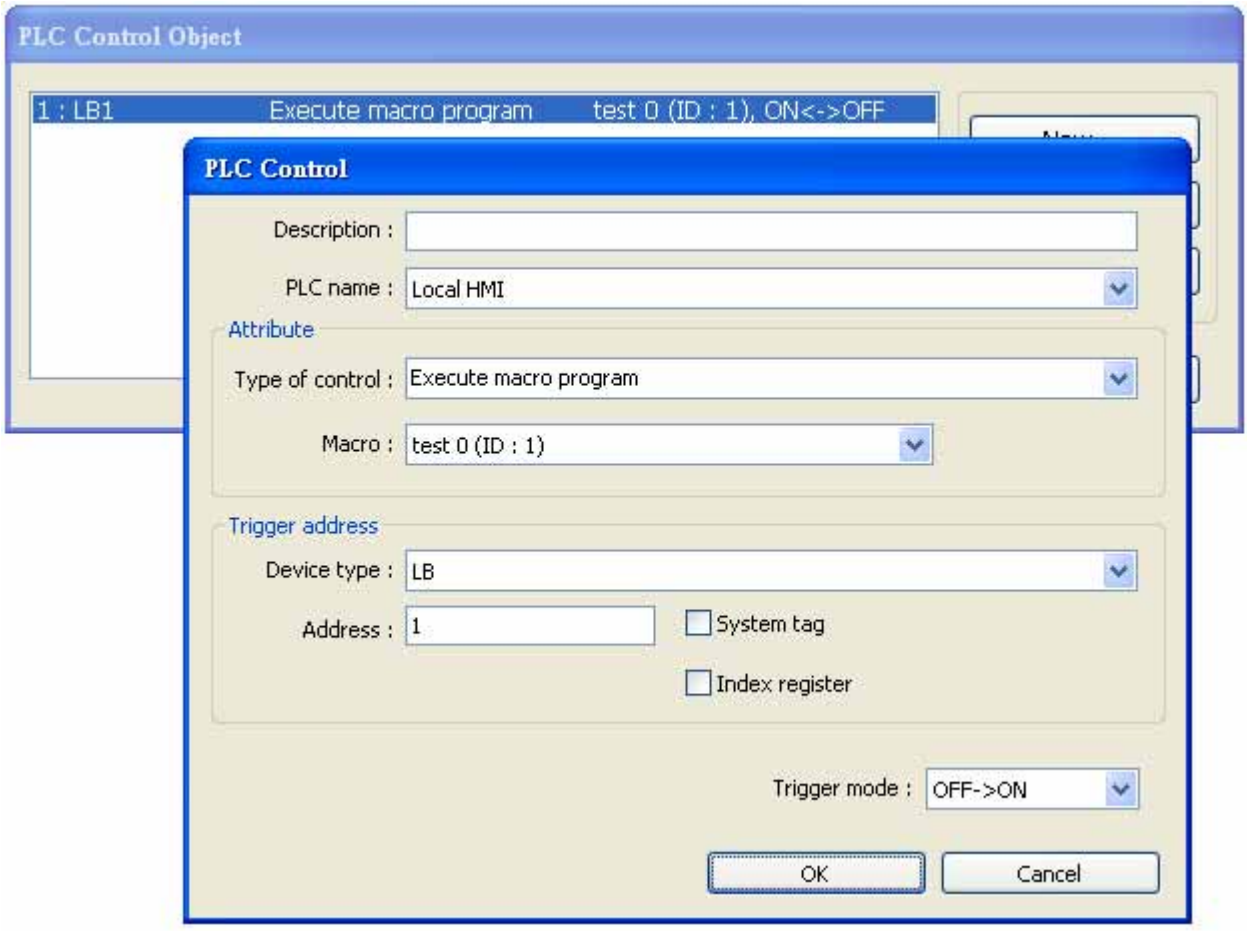
c. Delete

Highlight the selected function and push the button “Delete” to delete it.

3. Trigger condition of Macro

The objects of Set Bit, Toggle Switch, Function Key and PLC Control can use Macro, The following text will show how to trigger macro by use PLC Control object.

Step 1： Select control type to “Execute Macro Program” in the object property dialog of PlcControl.



Step 2：Select a Macro name and define a trigger condition in the object property dialog PlcControl(Now it is LB1).

- Some notes about using Macro

1. Limitation of storage space of Macro

The size of a Macro in a eob file is limited by the storage capacity. The maximum storage space of local variables in a Macro is 4K bytes. So the define range of different variable types are limited as following:

```
char a[4096]
bool b[4096]
short c[2048]
int   d[1024]
float e[1024]
```

2. Limitation of maximum lines of Macro to execute

There are at most 255 Macros in a eob file.

3. Macro may possibly cause deadlock of the machine.

When there is a infinite loop in a Macro without communicating with PLC

When the size of array exceeds the storage space in a Macro.

4. The Limitation of communication speed of Macro

The execution of Macro may be a little slow down when communicating with PLC.

This is caused by the data transferring time. Avoid too many complicated action in the Macro.

- Compiler error message

1. Error message format:

error C# : error description

is the error number.

Example: error C37 : undeclared identifier : i

When there are compile errors, the error description can be referenced by the compile error message number.

2：Error description

(C1) syntax error：'identifier'

There are many possibilities to cause compiler error.

For example :

```
macro_command main()
char i, 123xyz    //this is an unsupported variable name ,"Error message: "Syntax error: 123xyz"
end macro_command
```

(C2) 'identifier' used without having been initialized

Macro just support static array, must define the size of an array during declaration.

For example：

```
macro_command main()
char i
int g[i]       //    i used without having been initialized
end macro_command
```

(C3) redefinition error : 'identifier'

The name of variable and function within its scope must be unique.

For example：

```
macro_command main()
int g[10],   g      //error
end macro_command
```

(C4) function name error : ‘identifier’
reserved keywords and constant can not be the name of a function

For example：

```
sub int if()      // error
```

(C5) parentheses have not come in pairs
Statement missing “(“ or “)”

For example：

```
macro_command main    )//    missing C
```

(C6) illegal expression without matching ‘if’
Missing expression in If statement

(C7) illegal expression (no ‘then’) without matching ‘if’
Missing “Then” in If statement

(C8) illegal expression (no ‘end if’)
Missing “EndIf”

(C9) illegal ‘end if’ without matching ‘if’
Unfinished “If’ statement before “End If”

(C10) illegal ‘else’
The format of “If” statement is:
If [logic expression]Then

[Else [If [logic expression] Then]]

EndIf

Any format other than this format will cause compile error.

(C11) ‘case’ expression not constant
There should be constant behind “Case”

(C12) ‘select’ statement contains no ‘case’
Missing “Case” behind “Select”

(C13) illegal expression without matching ‘select case’
Missing “expression” behind “Select Case”

(C14) ‘select’ statement contains no ‘end select’
"Missing “End Select” statement

(C15) illegal ‘case’
Illegal “Case” statement"

(C16) illegal expression (no 'select') without matching ‘end select’
The format of “Select Case” statement is:
Select Case [expression]
Case [constant]

Case [constant]

Case [constant]

Case Else

End Select

Any format other than this format will cause compile error.

(C17) illegal expression (no 'for') without matching ‘next’
“For” statement error: missing “For” before “Next”

(C18) illegal variable type (not interger or char)
Should be integer of char variable

(C19) variable type error
Missing assign statement

(C20) must be key word ‘to’ or ‘down’
Missing keyword “to” or “down”
(C21) illegal expression (no 'next')

The format of “For” statement is:
For [variable] = [initial value] To [end value] [Step]

Next [variable]
Any format other than this format will cause compile error.

(C22) ‘wend’ statement contains no ‘while’
“While” statement error: missing “While” before “Wend”

(C23) illegal expression without matching ‘wend’

The format of “While” statement is:
While [logic expression]

Wend
Any format other than this format will cause compile error.

(C24) syntax error : ‘break’
“Break” statement can only be used in “For”, “While”, or “Select Case” statement
“Break” statement takes one line of Macro.

(C25) syntax error : ‘continue’
“Continue” statement can only be used in “For” statement, or “While” statement
“Continue” statement takes one line of Macro.

(C26) syntax error
expression is error.
(C27) syntax error
The mismatch of operation object in expression cause compile error.

For example：

```
macro_command main( )
int a, b

for a = 0 to 2
      b = 4 + xyz //illegal operation object
next a
end macro_command
```

(C28) must be ‘macro_command’
There must be ’macro_command’

(C29) must be key word ‘Sub’

The format of function declaration is:

```
sub [data type] function_name(…)
……..
end sub
```

For example::

```
sub int pow(int exp)
…….
end sub
```

Any format other than this format will cause compile error.

(C30) number of parameters is incorrect
Mismatch of the number of parameters

(C31) parameter type is incorrect
Mismatch of data type of parameter

(C32) variable is incorrect
The parameters of a function must be equivalent to the arguments passing to a function to avoid compile error.

(C33) function name : undeclared function
Undefined function

(C34) expected constant expression
Illegal member of array

(C35) invalid array declaration
Illegal definition of array

(C36) array index error
Illegal index of array

(C37) undeclared identifier : i ‘identifier’
Any variable or function should be declared before use.

(C38) PLC encoding method is not supported
The parameter of GetData(…) , SetData(…) should be legal PLC address.

(C39) ‘idenifier’ must be integer, char or constant
The format of array is:
Declaration: array_name[constant] (constant is the size of the array)
Usage: array_name[integer, character or constant]
Any format other than this format will cause compile error.

(C40) execution syntax should not exist before variable declaration or constant definition

For example :
Macro_Command main( )
int a, b

```
For a = 0 To 2
    b = 4 + a
int h , k
//declaration statement position error//
Next a
End Macro_Command
```

(C41) float variables cannot be contained in shift calculation
Floating point can not bitwise shift

(C42) function must return a value
Missing function return value

(C43) function should not return a value
Function can not return a value

(C44) float variables cannot be contained in calculation
Illegal Float data type in expression

(C45) PLC address error
Error PLC address

(C46) array size overflow (max. 4k)
Stack can not exceed 4k bytes

(C47) macro command entry function is not only one
Only one main entrance in the Macro is allowed

(C48) macro command entry function must be only one

The only one main entrance of Macro is:
Macro_Command function_name( )

End Macro_Command

(C49) a extended addresse's station no. must be between 0 and 255

For example:

SetData(bits[0] , “PLC 1”, LB , 300#123, 100)
300#123 中的 300 means the station no is 300, but the maximum is 255

(C50) a invalid PLC name
PLC name is not included in the Device List of system paramter
For example:

SetData(bits[0] , “PLC 1”, LB , 300#123, 100)

There is no “PLC 1” in Device List.

(C51) macro command do not control a remote device
Macro just can control local machine
For example

SetData(bits[0] , “PLC 1”, LB , 300#123, 100)

“PLC 1“ is connected with the remote device ,so it is can not work.

- Example source code

1:”For” statement and other expressions (arithmetic, bitwise shift, logic and comparison)

```
macro_command main()
int a[10], b[10], i

b[0]= (400 + 400 << 2) / 401
b[1]= 22 *2 - 30 % 7
b[2]= 111 >> 2
b[3]= 403 > 9 + 3 >= 9 + 3 < 4 + 3 <= 8 + 8 == 8
b[4]= not 8 + 1 and 2 + 1 or 0 + 1 xor 2
b[5]= 405 and 3 and not 0
b[6]= 8 & 4 + 4 & 4 + 8 | 4 + 8 ^ 4
b[7]= 6 – (~4)
b[8]= 0x11
b[9]= 409

for i = 0 to 4 step 1
    if (a[0] == 400) then
        GetData(a[0],”Device 1”, 3x, 0,9)
        GetData(b[0],”Device 1”, 3x, 11,10)
end If
next i
end macro_command
```

2.while, if, break

```
macro_command main()
int b[10], i
i = 5
while i == 5 - 20 % 3
        GetData(b[1], ”Device 1”, 3x, 11, 1)

        if b[1] == 100 then
            break
        end if
```

```
wend
end macro_command
```

3:Global variables and function call

```
char g

sub int fun(int j, int k)
int y

SetData(j, “Local HMI”, LB, 14, 1)
GetData(y, “Local HMI”, LB, 15, 1)
g = y

return y
end Sub

macro_command main()
int a, b, i

a = 2
b = 3
i = fun(a, b)
SetData(i, “Local HMI”, LB, 16, 1)
end macro_command
```

4. ”If” statement

```
macro_command main()
int k[10], j

for j = 0 to 10
    k[j] = j
next j

if k[0] == 0 then
    SetData(k[1], “Device 1”, 3x, 0, 1)
end if
```

```
if k[0] == 0 then
    SetData(k[1], “Device 1”, 3x, 0, 1)
else
        SetData(k[2], “Device 1”, 3x, 0, 1)
end if

if k[0] == 0 then
    SetData(k[1], “Device 1”, 3x, 1, 1)
else if k[2] == 1 then
    SetData(k[3], “Device 1”, 3x, 2, 1)
end If

if k[0] == 0 then
    SetData(k[1], “Device 1”, 3x, 3, 1)
else if k[2] == 2 then
    SetData(k[3], “Device 1”, 3x, 4, 1)
else
    SetData(k[4], 3x_BIN, 5, 1)
end If
end macro_command
```

5. while statement

```
macro_command main()
char i = 0
int a[13], b[14], c = 4848

b[0] = 13

while b[0]
    a[i] = 20 + i * 10

    if a[i] == 120 then
        c =200
        break
    end if
```

```
        i = i + 1
    wend

    SetData(c, “Device 1”, 3x, 2, 1)
    end macro_command
```

6. break、continue statement

```
    macro_command main()
    char i = 0
    int a[13], b[14], c = 4848

    b[0] = 13

    while b[0]
        a[i] = 20 + i * 10

        if a[i] == 120 then
            c =200
            i = i + 1
            continue
        end if

        i = i + 1

        if c == 200 then
            SetData(c, “Device 1”, 3x, 2, 1)
        break
        end if
    wend
    end macro_command
```

7. array statement

```
    macro_command main()
    int a[25], b[25], i

    b[0] = 13
```

```
for i = 0 to b[0] step 1
        a[i] = 20 + i * 10
next i

SetData(a[0], “Device 1”, 3x, 0, 13)
end macro_command
```